# Observational Proofs by Implicit Context Induction

Narjes Berregeb, Adel Bouhoula, Michaël Rusinowitch

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

# *Observational Proofs by Implicit Context Induction*

Narjes Berregeb  Adel Bouhoula  Michaël Rusinowitch

**N ° 3151**

Avril 1997

THÈME 2


ISSN 0249-6399

# Observational Proofs by Implicit Context Induction

Narjes Berregeb * Adel Bouhoula † Michaël Rusinowitch ‡

Thème 2 — Génie logiciel
et calcul symbolique
Projet PROTHEO

Rapport de recherche n ° 3151 — Avril 1997 — 35 pages

**Abstract:** Observability concepts contribute to a better understanding of software correctness. In order to prove observational properties, the powerful concept of *Context Induction* has been developed by Hennicker [Hen91]. We propose in this paper to embed Context Induction in the implicit induction framework of [BR95]. The proof system we obtain applies to conditional specifications. It allows for many rewriting techniques and for the refutation of false conjectures. Under reasonable assumptions it is refutationally complete. Moreover this proof system is operational: it has been implemented within the Spike prover and interesting computer experiments are reported.

**Key-words:** Observability, Context induction, Implicit induction, Conditional Specification

*(Résumé : tsvp)*

* E-mail: berregeb@loria.fr
† E-mail: bouhoula@loria.fr
‡ E-mail: rusi@loria.fr

Unité de recherche INRIA Lorraine
Technopôle de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY (France)
Téléphone : (33) 83 59 30 30 – Télécopie : (33) 83 27 83 19
Antenne de Metz, technopôle de Metz 2000, 4 rue Marconi, 55070 METZ
Téléphone : (33) 87 20 35 00 – Télécopie : (33) 87 76 39 77

# Preuves Observationnelles par Induction de Contextes Implicite

**Résumé :** Les concepts d'observabilité contribuent à l'amélioration de la notion de correction de logiciels. Dans le but de prouver des propriétés observationnelles, un nouveau concept d'*induction de contextes* a été développé par Hennicker [Hen91]. Nous proposons d'intégrer l'induction de contextes dans le cadre de l'induction implicite de [BR95]. Le système de preuve obtenu s'applique des spécifications conditionnelles. Il utilise diverses techniques de réécriture et permet la réfutation des conjecture non valides. Sous des hypothèses raisonnables, il est réfutationnellement complet. Ce système de preuve est opérationnel: il a été implémenté dans le prouveur Spike et les premières expérimentations sont prometteuses.

**Mots-clé :** Observabilité, Induction de contextes, Induction implicite, Spécification conditionnelles

# 1 Introduction

Observational concepts are fundamental in formal methods since for proving the correctness of a program with respect to a specification it is essential to be able to abstract away from internal implementation details. Data objects can be viewed as equal if they cannot be distinguished by experiments with observable result. The idea that the semantic of a specification must describe the *behaviour* of an abstract data type as viewed by an extern user, is du to [Gut75, GGM76]. Though a lot of work has been devoted to the semantical aspects of observability [ST85, NO87, Hen89, BB91, BBK92, Pad96] (see [BBK91] for a classification), few proof techniques have been studied [ST89, Sch90, Hen91, MG94, Lys94, BH96], and even less have been implemented. In this paper, we propose an automatic method for proving observational properties of conditional specifications. The method relies on computing families of well chosen contexts, called *test contexts*, that "cover" in some sense all observable ones. These families are applied as induction schemes. Our inference system basically consists in extending terms by test contexts and simplifying the results with a powerful rewriting machinery in order to generate new subgoals. An advantage of this approach is that it allows also for disproving false observational conjectures. The method is even refutationally complete for an interesting class of specifications. From a preliminary implementation on top of the Spike prover computer experiments are reported. The given examples have been treated in a fully automatic way by the program.

# 2 Related works

Hennicker [Hen91] has proposed an induction principle, called *context induction*, which is a proof principle for behavioural abstractions. A property is observationally valid if it is valid for all observable *experiments*. Such experiments are represented by *observable contexts*, which are context of observable sort over the signature of a specification where a distinguished subset of its sorts is specified as observable. Hence, a property is valid for all observable experiments if it is valid for all corresponding observable contexts. A context $c$ is viewed as a particular term containing exactly one variable; therefore, the subterm ordering defines a noetherian relation on the set of observable contexts. Consequently, the principle of structural induction induces a proof principle for properties of contexts of observable sort, which is called *context induction*. This approach provides with a uniform proof method for the verification of behavioural properties. It has been implemented in the system ISAR [BH93]. However, in concrete examples, this verification is a non trivial task, and requires human guidance: the system often needs a generalization of the current induction assertion before each nested context induction, so that to achieve the proof.

Malcolm and Goguen [MG94] have proposed a proof technique which simplifies Hennicker proofs. The idea is to split the signature into *generators* and *defined functions*. Proving that two terms are behaviourally equivalent, comes to prove that they give the same result in all observable contexts built from defined functions, provided that the generators verify a

congruence relations w.r.t. behavioural equivalence. This proof technique is an efficient optimization of Hennicker proofs, however, explicit context induction is not avoided.

Bidoit and Henniker [BH92] have investigated how a first order logic theorem prover can be used to prove properties in an observational framework. The method consists in computing automatically some special contexts called *crucial contexts*, and in enriching the specification so that to automatically prove observational properties. But this method was only developed for the proof of equations and for specifications where only one sort is not observable. Besides, it fails on several examples (cf. stack example), where it is not possible to compute crucial contexts.

Bidoit and Hennicker [BH96] have also investigated characterization of behavioural theories that allows for proving behavioural theorems with standard proofs techniques for first order logic. In particular they propose general conditions under which an infinite axiomatization of the observational equality can be transformed into a finitary one. However, in general there is no automatic procedure for generating such a finite axiomatization of the observational equality.

Lysne [Lys94] has developed a method for proof by consistency w.r.t. the final model. Lysne extends Bachmair's method for proof by consistency to the final algebra framework. The proof technique is based on a special completion procedure whose idea is to consider, not only critical pairs emerging from positioning rewrite rules on equations, but also those emerging from positioning equations on to rewrite rules. This approach is restricted to equations and requires the ground convergence property of the axioms in order to be sound (in our case, ground convergence is needed only for completeness).

# 3 Basic notions

We assume that the reader is familiar with the basic concepts of algebraic specifications [Wir90], term rewriting and equational reasoning. A many sorted signature $\Sigma$ is a pair $(S, F)$ where $S$ is a set of *sorts* and $F$ is a set of function symbols. For short, a many sorted signature $\sigma$ will simply be denoted by $F$. We assume that we have a partition of $F$ in two subsets, the first one, $\mathcal{C}$, contains the *constructor symbols* and the second, $\mathcal{D}$, is the set of *defined symbols*. Let $X$ be a family of sorted variables and let $T(F, X)$ be the set of sorted terms. $var(t)$ stands for the set of all variables appearing in $t$. A term is *linear* if all its variables occur only once in it. If $var(t)$ is empty then $t$ is a *ground* term. The set of all ground terms is $T(F)$. Let $A$ be an arbitrary non-empty set, and let $F_A = \{f_A \mid f \in F\}$ such that if $f$ is of arity $n$ then $f_A$ is a function from $A^n$ to $A$. The pair $(A, F)$ is called a $\Sigma$-*algebra*, and $A$ the *carrier* of the algebra. For sake of simplicity, we will write $A$ to denote the $\Sigma$-algebra when $F$ and $F_A$ are obvious. We denote by $eval_A$ the homomorphism from $T(F)$ to $A$ which evaluates a ground term in $A$. The $\Sigma$-algebra $A$ is *term-generated* if $eval_A$ is surjective. A *valuation* $\theta$ is an application from $X$ to $A$. It uniquely extends a homomorphism from $T(F, X)$ to $A$. If $t$ is a term, then $t\theta$ denotes the application of $\theta$ to $t$. An algebra $A$ is *initial* in a class of algebras if it is in the class and there is one and only one homomorphism from $A$ to any other algebra in the class. An algebra $A$ is *final* in a

class of algebras if it is in the class and there is one and only one homomorphism from every algebra in the class to $A$. A *substitution* assigns terms of appropriate sorts to variables. The domain of $\eta$ is defined by: $dom(\eta) = \{x \mid x\eta \neq x\}$. If $\eta$ applies every variable to a ground term, then $\eta$ is a ground substitution. We denote by $\equiv$ the syntactic equivalence between objects. Let $\mathbb{N}^*$ be the set of sequences of positive integers. For any term $t$, $Pos(t) \subseteq N^*$ denotes its set of positions and the expression $t/u$ denotes the *subterm of $t$ at a position $u$*. We write $t[s]_u$ (resp. $t[s]$ ) to indicate that $s$ is a subterm of $t$ at position $u$ (resp. at some position).The top position is written $\varepsilon$. Let $t(u)$ denote the symbol of $t$ at position $u$. A position $u$ in a term $t$ is said to be *a strict position* if $t(u) = f \in F$. A position $u$ in a term $t$ such that $t(u) = x$ and $x \in X$, is a *linear variable position* if $x$ occurs only once in $t$, otherwise, $u$ is a *non linear variable position*. The depth of a term $t$ is defined as follows: $|t| = 0$ if $t$ is a constant or a variable, otherwise, $|f(t_1, \ldots, t_n)| = 1 + max_i|t_i|$. We denote by $\succ$ a transitive irreflexive relation on the set of terms, that is noetherian, monotonic ($s \succ t$ implies $w[s]_u \succ w[t]_u$), stable per instantiation ($s \succ t$ implies $s\sigma \succ t\sigma$) and satisfies the subterm property ($f(\cdots, t, \cdots) \succ t$). If two terms $s$ and $t$ are incomparable, we denote it by $s \not\succ t$. The multiset extension of $\succ$ will be denoted by $\gg$. An *equation* is a formula of the form $l = r$. A *conditional* equation is a formula of the following form: $\wedge_{i=1}^n a_i = b_i \Rightarrow l = r$. It will be written $\wedge_{i=1}^n a_i = b_i \Rightarrow l \rightarrow r$ and called a *conditional rule* if $\{l\sigma\} \gg \{r\sigma, a_1\sigma, b_1\sigma, \cdots, a_n\sigma, b_n\sigma\}$ for each substitution $\sigma$. The *precondition* of rule $\wedge_{i=1}^n a_i = b_i \Rightarrow l \rightarrow r$ is $\wedge_{i=1}^n a_i = b_i$. The term $l$ is the *left-hand side* of the rule. A rewrite rule $c \Rightarrow l \rightarrow r$ is *left-linear* if $l$ is linear. A set of conditional rules is called a *rewrite system*. A constructor is *free* if it is not the root of a left-hand side of a rule. A rewrite system $R$ is *left-linear* if every rule in $R$ is left-linear. We define $depth(R)$ as the maximal depth of the strict positions in its left-hand sides. Let $R$ be a set of conditional rules. Let $t$ be a term and $u$ a position in $t$. We write: $t[l\sigma]_u \rightarrow_R t[r\sigma]_u$ if there is a substitution $\sigma$ and a conditional equation $\wedge_{i=1}^n\ a_i = b_i \Rightarrow l = r$ in $R$ such that:

1. $l\sigma \succ r\sigma$.
2. for all $i \in [1 \cdots n]$ there exists $c_i$ such that $a_i\sigma \rightarrow_R^* c_i$ and $b_i\sigma \rightarrow_R^* c_i$.
3. $\{t[l\sigma]_u\} \gg \{a_1\sigma,\ b_1\sigma,\ \cdots,\ a_n\sigma,\ b_n\sigma\}$.

A term $t$ is *irreducible* (or in *normal form*) if there is no term $s$ such that $t \rightarrow_R s$. A term $t$ is *ground reducible* iff all its ground instances are reducible. A symbol $f \in F$ is *completely defined* if all ground terms with root $f$ are reducible to terms in $T(\mathcal{C})$. We say that $R$ is *sufficiently complete* if all symbols in $\mathcal{D}$ are completely defined. A *clause* $C$ is an expression of the form : $\wedge_{i=1}^n a_i = b_i \Rightarrow \vee_{j=1}^m a'_j = b'_j$. We denote by $A \models C$, the validity of $C$ in $A$. Let $E$ be a set of conditional equations. An algebra $A$ is an *(F,E)-model* iff $A$ satisfies all conditional equations in $E$. We denote by $Mod(F, E)$ the class of term-generated algebras that are (F,E)-models. The clause $C$ is a *logical consequence* of $E$ if $C$ is valid in any model of $E$, denoted, by abuse of notation, by $E \models C$. An initial algebra in $Mod(F, E)$ is known to exist, is unique up to isomorphism, and will be denoted by $I(F, E)$. We say that $C$ is *inductively valid* in $E$ and denote it by $E \models_{ind} C$ if $I(F, E) \models C$. By corollary 4.3.3 of [Pad88], it is equivalent to: for any ground substitution $\sigma$, (for all $i$, $E \models a_i\sigma = b_i\sigma$) implies (there exists $j$, $E \models a'_j\sigma = b'_j\sigma$). We say that two terms $s$ and $t$ are joinable, denoted by

$s \downarrow_R t$, if $s \rightarrow_R^* v$ and $t \rightarrow_R^* v$ for some term $v$. The rewrite system $R$ is *ground convergent* if the terms $u$ and $v$ are joinable whenever $u, v \in T(F)$ and $R \models u = v$.

# 4 Observational semantics

The notion of *observation technique* have been introduced as a mean for describing what is observed in a given algebra. Various techniques have been proposed: observations based on sorts [Wan79, Rei84, NO87, Hen91], operators[BB91], terms [ST88, Hen89, BBK92] or formula [ST85, ST88, Kna91]. The observation technique we use in our method is based on sorts[1]. The semantic we choose is based on a relaxing of the satisfaction relation. The notion of *context* is fundamental in all approaches based on such observational semantics. An observational property is obtained by taking into account only observable information. Thus, to show that it is valid, one has to show its validity in all *observable contexts*.

**Definition 4.1 (context)** *Let $T(F, X)$ be a term algebra and $(S, F)$ be its signature.*

- *a* context *over $F$ is a non-ground term $c \in T(F, X)$ with a distinguished occurrence of a variable called the* context variable *of $c$. To indicate the context variable $z_s$ occuring in $c$, we often write $c[z_s]$ instead of $c$, where $s$ is the sort of $z_s$.*
- *a context reduced to a variable $z_s$ of sort $s$ is called an* empty context *of sort $s$.*
- *the application of a context $c[z_s]$ to a term $t \in T(F, X)$ of sort $s$, denoted by $c[t]$, is defined by the substitution of $z_s$ by $t$ in $c[z_s]$. The context $c$ is said to be* applicable *to $t$.*
- *by exception, $var(c)$ will denote the set of variables occurring in $c$ but the context variable of $c$. A context $c$ is* ground *if $var(c) = \emptyset$. We denote by $|c|$ the depth of $c$.*
- *a* subcontext *(resp.* strict subcontext*) of $c$, is a subterm (resp. strict subterm) of $c$ with the same contextual variable.*

**Notations**
Let $c[z_s]$ and $c'[z'_{s'}]$ be contexts such that $c'$ is of sort $s$, let $t$ be a term and $\sigma$ be a substitution such that $z_s \notin dom(\sigma)$. We use the following notations:

- $c[(c'[t])] = (c[c'])[t] = c[c'[t]]$
- $(c[t])\sigma = (c\sigma)[t\sigma] = c[t]\sigma$

**Definition 4.2 (specification, observational specification)** *A specification $SP$ is a triple $(S, F, E)$ where $(S, F)$ is a signature and $E$ is a set of conditional equations. An* observational specification *$SP_{obs}$ is a couple $(SP, S_{obs})$ such that $SP = (S, F, E)$ is a specification and $S_{obs} \subseteq \mathcal{S}$ is the set of* observable sorts.

[1] but it can be easily extended to observations based on operators

**specification:** STACK
**sorts:** nat, stack
**observable sorts:** nat
**constructors:**
0: →nat;
$s$: nat →nat;
$Nil$: →stack;
$push$: nat × stack →stack;
**defined functions:**
$top$: stack → nat;
$pop$: stack → stack;
**axioms:**
$top(Nil) = 0$
$top(push(i, s)) = i$
$pop(Nil) = Nil$
$pop(push(i, s)) = s$

Figure 1: Stack specification

In the following, we denote by $SP_{obs} = (SP, S_{obs})$ an observational specification, where $SP = (S, F, E)$. We also denote by $C \equiv \wedge_{i=1}^{n} a_i = b_i \Rightarrow \vee_{j=1}^{m} a'_j = b'_j$, a clause.

**Example 4.1** *The Stack specification in figure 1, is an observational specification, where* $S_{obs} = \{nat\}$.

**Definition 4.3 (observable context, term, equation, precondition)** *An observable term is a term whose sort belongs to* $S_{obs}$. *The set of observable contexts is denoted by* $\mathcal{C}_{obs}$ . *An equation* $a = b$ *is observable if* $a$ *and* $b$ *are observable. The precondition of a rule is observable if all its equations are observable.*

**Example 4.2** *Consider the specification stack in figure 1. They are infinitely many observable contexts:* $top(z_{stack}), top(pop(z_{stack})), \ldots, top(pop(\ldots(pop(z_{stack}))\ldots)), \ldots$
$top(push(z_{stack})), top(push(i, pop(z_{stack}))), \ldots$

The notion of observational validity is based on the idea that two objects in a given algebra are observationally valid if they cannot be distinguished by computations with observable results. These computations are formalized by contexts.

**Definition 4.4 (observable equality in an algebra)** *Let* $A$ *be an algebra, and* $a, b$ *be two elements of* $A$. *We say that* $a$ *and* $b$ *are observationally equal in* $A$, *and we denote it by* $A \models_{obs} a = b$ *iff: for all observable contexts* $c[z_s]$, *for all valuations* $\theta_a, \theta_b$ *such that* $z_s\theta_a = a$ *and* $z_s\theta_b = b$, $A \models c\theta_a = c\theta_b$.

**Definition 4.5 (observable satisfaction)** *Let $A$ be an algebra. We say that $C$ is observationally valid in $A$ and we denote it by $A \models_{obs} C$, iff: for all valuation $\theta$, if (for all $i \in [1..n]$, $A \models_{obs} a_i\theta = b_i\theta$) then (there exists $j \in [1..m]$, $A \models_{obs} a'_j\theta = b'_j\theta$)*

**Definition 4.6 ( $=_{obs}$ )** *Let $a$ and $b$ be two terms. We say that $a$ and $b$ are observationally equal, and we denote it by $E \models_{obs} a = b$ or by $a =_{obs} b$ iff for all $c \in \mathcal{C}_{obs}$, $E \models_{ind} c[a] = c[b]$.*

**Example 4.3** *Consider the Stack specification in figure 1. It is easy to see that $push(top(s), pop(s)) = s$ is not satisfied (in the classical sense), because $push(top(Nil), pop(Nil)) = Nil$ is not valid. However, intuitively, it is observationally satisfied if we just* observe *the elements of the sequences $push(top(s), pop(s))$ and $s$. This can be formally shown by considering all observable contexts.*

The following lemma ties the concept of observational validity to initial model.

**Lemma 4.1** *Let $a$ and $b$ be two terms. Then:*

$$E \models_{obs} a = b \;\Leftrightarrow\; I(F,E) \models_{obs} a = b$$

**Proof:**

$$\begin{array}{lll} E \models_{obs} a = b & \Leftrightarrow & \forall c \in \mathcal{C}_{obs},\ E \models_{ind} c[a] = c[b] \text{ (Definition 4.6)} \\ & \Leftrightarrow & I(F,E) \models c[a] = c[b] \text{ (Definition of an inductive theorem)} \\ & \Leftrightarrow & \forall \sigma\ ground,\ I(F,E) \models c[a]\sigma = c[b]\sigma \\ & \Leftrightarrow & I(F,E) \models_{obs} a = b \text{ (Definition 4.4)} \end{array}$$

□

**Lemma 4.2** *The relation $=_{obs}$ is a congruence on $T(F)$*

**Proof:** The relation $=_{obs}$ is obviously an equivalence relation on $T(F)$. Let $a_1, \ldots, a_n, b_1, \ldots, b_n \in T(F)$, and $f : s_1 \times \ldots \times s_n \to s$. Suppose $a_i =_{obs} b_i$ for all $i \in [1..n]$, and let us show that $f(a_1, .., a_n) =_{obs} f(b_1, \ldots, b_n)$. We will show that for all $j \in [1..n]$, if $a_i =_{obs} b_i$, then $f(a_1, \ldots, a_j, x_{j+1}, \ldots, x_n) =_{obs} f(b_1, \ldots, b_j, x_{j+1}, \ldots, x_n)$ (by induction on $j$). Then, by an induction argument, we conclude that if $a_i =_{obs} b_i$ for all $i \in [1..n]$, then $f(a_1, \ldots, a_n) =_{obs} f(b_1, \ldots, b_n)$.

- $j = 1$, we have $a_1 =_{obs} b_1$. Let $c[z_s] \in \mathcal{C}_{obs}$ , then $c[f(z_{s_1}, x_2, \ldots, x_n)] \in \mathcal{C}_{obs}$ . So, $E \models_{ind} c[f(a_1, x_2, \ldots, x_n)] = c[f(b_1, x_2, \ldots, x_n)]$. Thus, $f(a_1, x_2, \ldots, x_n) =_{obs} f(b_1, x_2, \ldots, x_n)$.
- $j > 1$, let $c[z_s] \in \mathcal{C}_{obs}$ . Suppose $E \models_{ind} c[f(a_1, \ldots, a_i, z_{i+1}, x_{i+2}, \ldots, x_n) = c[f(b_1, \ldots, b_i, z_{i+1}, x_{i+2}, \ldots, x_n)]$. Then, since $a_{i+1} =_{obs} b_{i+1}$, we have $E \models_{ind} c[f(a_1, \ldots, a_i, a_{i+1}, x_{i+2}, \ldots, x_n) = c[f(b_1, \ldots, b_i, b_{i+1}, x_{i+2}, \ldots, x_n)]$

Hence, for all $c \in \mathcal{C}_{obs}$ , if $a_i =_{obs} b_i$ for all $i \in [1..n]$, then $E \models_{ind} c[f(a_1, \ldots, a_n)] = c[f(b_1, \ldots, b_n)]$. Thus, if $a_i =_{obs} b_i$ for all $i \in [1 \ldots n]$, then $f(a_1, \ldots, a_n) =_{obs} f(b_1, \ldots, b_n)$. □

Our aim is to generalize implicit induction proofs, to an observational framework. In this way, all inductive theorems will be also observational theorems. In particular, if $S_{obs} = S$, we recover the proofs in the initial model. However, this generalization is not straightforward since we deal with conditional specifications: Let $u = v$ and $u' = v'$ two ground equations, and suppose that $E \not\models_{ind} u = v$ and $E \not\models_{ind} u' = v'$. We may have $E \models_{obs} u' = v'$ and $E \not\models_{obs} u = v$. In this case, $E \models_{ind} u' = v' \Rightarrow u = v$ but $E \not\models_{obs} u' = v' \Rightarrow u = v$.

For this reason, we use a semantic close to the one in [Pad88]. The models $A$ we are concerned with in our observational semantics are term-generated. Besides, they are *visibly initial* [Pad88]: for all observable equation $e$, $A \models e$ implies $E \models_{ind} e$.

**Theorem 4.1** *Let $A$ be a visibly initial algebra, and $t, t'$ two ground terms.*

$$A \models t = t' \quad implies \quad E \models_{obs} t = t'$$

**Proof:** We have $A \models t = t'$, then for all $c \in \mathcal{C}_{obs}$ , $A \models c[t] = c[t']$. By visible initiality of $A$, for all $c \in \mathcal{C}_{obs}$ , $E \models_{ind} c[t] = c[t']$. Therefore, $E \models_{obs} t = t'$. □

**Definition 4.7 (observational class)** *The observational class $Obs(SP_{obs})$ is the class of term-generated and visibly initial algebras $A$ such that $\forall e \in E, A \models_{obs} e$.*

The follwing theorem, similar to lemma 4.6.1 in [Pad88], states that the class $Obs(SP_{obs})$ has a final algebra.

**Theorem 4.2** *Suppose that all the preconditions of $E$ are observable and let $T(F, E)$ denote the quotient algebra of $T(F)$ w.r.t. $=_{obs}$. Then, $T(F, E)$ is a final algebra in $Obs(SP_{obs})$.*

**Proof:** $T(F, E)$ is visibly initial: suppose that $T(F, E) \models u = v$, such that $u$ and $v$ are observable. Then, for all ground substitution $\sigma$, $T(F, E) \models u\sigma = v\sigma$. So, $E \models_{obs} u\sigma = v\sigma$. Since $u$ and $v$ are observable, we have $E \models_{ind} u\sigma = v\sigma$. Then, $E \models_{ind} u = v$.

Let $e \equiv \wedge_{i=1}^{n} u_i = v_i \Rightarrow s = t$ be a conditional equation in $E$. Suppose that for all ground substitutions $\sigma$, $T(F, E) \models_{obs} u_i\sigma = v_i\sigma$, for all $i \in [1..n]$. Since $u_i, v_i$ for all $i \in [1..n]$ are observable, then $T(F, E) \models u_i\sigma = v_i\sigma$. By construction of $T(F, E)$, we have $u_i\sigma =_{obs} v_i\sigma$. On the other hand, $E \models_{ind} e$, by definition of the initial model. Then, by definition of $=_{obs}$, $s\sigma =_{obs} t\sigma$. So, $T(F, E) \models s\sigma = t\sigma$, consequently $T(F, E) \models_{obs} s\sigma = t\sigma$. Thus, $T(F, E) \in Obs(SP_{obs})$.

Let $A \in Obs(SP_{obs})$, and *nat* be a homomorphism from $T(F)$ in $T(F, E)$, which assigns to $t \in T(F)$, the set of $t' \in T(F)$ such that $t =_{obs} t'$. Let $eval_A$ be the evaluation morphism of $A$. We define $abs_A$ of $A$ in $T(F, E)$ by $abs_A \circ eval_A = nat$. Thanks to Theorem 4.1,

$abs_A$ is well defined. Let us show that $abs_A$ is a homomorphism: let $a \in A$. Since $A$ is term generated, there exists a ground term $t$ such that $eval_A(t) = a$. We obtain:

$$\begin{aligned} abs_A(f_A(a)) &= abs_A \circ f_A \circ eval_A(t) \\ &= abs_A \circ eval_A(f(t)) \\ &= nat(f(t)) \\ &= f_{T(F,E)} \circ nat(t) \\ &= f_{T(F,E)} \circ abs_A \circ eval_A(t) \\ &= f_{T(F,E)} \circ abs_A(a) \end{aligned}$$

Since every homomorphism $h$ from $A$ to $T(F, E)$ must fulfill $h \circ eval_A = nat$, then $abs_A$ is unique. □

**Theorem 4.3** *Suppose that all the preconditions of $E$ are observable. Then, $I(F, E)$ is an initial algebra in $Obs(SP_{obs})$.*

**Proof:** $I(F, E)$ is an initial algebra in $Mod(F, E)$. Then, it is sufficient to show that $I(F, E)$ is an algebra in $Obs(SP_{obs})$.

- $I(F, E)$ is visibly initial: by definition of an inductive theorem.
- $I(F, E)$ is an observational model of $E$: Let $e \equiv \wedge_{i=1}^{n} u_i = v_i \Rightarrow s = t$ a conditional equation of $E$. Let $\sigma$ be a ground substitution $\sigma$ such that $I(F, E) \models_{obs} u_i\sigma = v_i\sigma$, then $I(F, E) \models u_i\sigma = v_i\sigma$ since $u_i$ and $v_i$ are observable. Then $I(F, E) \models s\sigma = t\sigma$, since $I(F, E)$ is an (F,E)-model. Per consequent, $I(F, E) \models_{obs} s\sigma = t\sigma$.

Thus, $I(F, E) \in Obs(SP_{obs})$. □

**Theorem 4.4** *Suppose that all the preconditions of $E$ are observable. Then $T(F, E) \models C$ iff: for all ground substitutions $\sigma$, if (for all $i, E \models_{obs} a_i\sigma = b_i\sigma$) then (there exists $j, E \models_{obs} a'_j\sigma = b'_j\sigma$)*

**Proof:**. The clause $C$ is valid in $T(F, E)$ iff: for all valuation $\theta$, $T(F, E) \models a_i\theta = b_i\theta$, for all $i \in [1..n]$, there exists $j \in [1..m]$, $T(F, E) \models a'_j\theta = b'_j\theta$. By construction of $T(F, E)$, we deduce that for all ground substitution $\sigma$, if $T(F, E) \models a_i\sigma = b_i\sigma$, for all $i \in [1..n]$, then there exists $j \in [1..m]$ such that $T(F, E) \models a'_j\sigma = b'_j\sigma$. Since all the equations are ground, it is equivalent to say that: if (for all $i$, $a_i\sigma =_{obs} b_i\sigma$) then (there exists $j$ such that $a'_j\sigma =_{obs} b'_j\sigma$). Thus, it is equivalent to say that: if (for all $i$, $E \models_{obs} a_i\sigma = b_i\sigma$) then (there exists $j$ such that $E \models_{obs} a'_j\sigma = b'_j\sigma$). □

We will also denote $T(F, E) \models C$ by $E \models_{obs} C$, where $C$ is a clause.

# 5 Induction schemes

Our purpose in this section is to introduce the ingredients allowing us to prove and disprove behavioural properties. This task amounts in general to check an infinite number of ground formulas for validity, since an infinite number of instances and an infinite number of contexts have to be considered for building these ground instances. This is where induction comes into play. Test substitutions will provide us with induction schemes for substitutions and test contexts will provide us with induction schemes for contexts. In general, it is not possible to consider all the observable contexts. However, cover contexts are sufficient to prove behavioural theorems by reasoning on the ground irreducible observable contexts rather than on the whole set of observable contexts. In the following, we denote by $R$ a conditional rewriting system and by $C \equiv \wedge_{i=1}^{n} a_i = b_i \Rightarrow \vee_{j=1}^{m} a'_j = b'_j$ a clause.

**Definition 5.1 (cover set)** *A* cover set*, denoted by $CS$, for $R$, is a finite set of irreducible terms such that for all ground irreducible term $s$, there exists a term $t$ in $CS$ and a ground substitution $\sigma$ such that $t\sigma \equiv s$.*

**Definition 5.2 (cover context, cover context set)** *A* cover context *$c$ is a context of sort $s$ such that there exists an irreducible ground observable context $c_{obs}[z_s]$ and $c_{obs}[c]$ is observable and irreducible. A* cover context set *$CC$ is a set of cover contexts such that: for each ground irreducible context $c_{obs}[z_s] \in \mathcal{C}_{obs}$ , there exists $c[z_s] \in CC$ and a substitution $\theta$ such that $dom(\theta) = var(c)$ and $c\theta$ is a subcontext of $c_{obs}$ .*

**Example 5.1** *A cover context set for the specification stack is $\{z_{nat}, top(z_{stack}), pop(z_{stack})\}$. The context $push(i, z_{stack})$ is not a cover context since $top(push(i, z_{stack}))$ and $pop(push(i, z_{stack}))$ are reducible. Note that usually there are infinitely many possible cover context sets. For instance, $\{z_{nat}, top(z_{stack}), top(pop(z_{stack})), pop(pop(z_{stack}))\}$ is also a cover context set.*

In the following, we refine test context sets so that to be able not only to prove behavioural properties, but also to disprove the non valid ones.

**Definition 5.3 (quasi ground reducibility)** *A context $c$ is* quasi ground reducible *if for all ground substitution $\tau$ such that $dom(\tau) = var(c)$, $c\tau$ is reducible.*

**Definition 5.4 (strongly irreducible)** *A term $t$ is* strongly irreducible *if none of its non-variable subterms matches a left-hand side of a rule in $R$. A positive clause $C_{pos}$ is* strongly irreducible *if for all $i \in [1..n]$, $a_i \not\equiv b_i$, and the maximal elements of $\{a_i,\ b_i\}$ w.r.t. $\prec$ are strongly irreducible by $R$.*

**Definition 5.5 (induction positions, induction variables)** *Let $f$ be a function symbol, we define the set of induction positions of $f$ as follows: $ind\_pos(f) = \{u \mid$ there is $p \Rightarrow g \rightarrow d \in R$ such that $g(\varepsilon) \equiv f$ and $u$ is either a strict position or a non-linear variable position in $g\}$. We say that $x$ is an* induction variable *of $t$ if $x$ occurs at a position $u.v$ of $t$ such that $v$ is an induction position of $t(u)$.*

**Definition 5.6 (clausal context)** *A clausal context $c$ for a clause $C$ is a set of contexts $< c_1, \ldots, c_n, c'_1, \ldots, c'_m >$ such that for all $i \in [1..n]:$ $c_i$ is applicable to $a_i = b_i$, and for all $j \in [1..m]:$ $c'_j$ is applicable to $a'_j = b'_j$. The application of $c$ to $C$, denoted by $c[C]$, gives the clause $\vee_{i=1}^{n} c_i[a_i] = c_i[b_i] \Rightarrow \wedge_{j=1}^{m} c'_j[a'_i] = c'_i[b'_i]$.*

Cover sets and cover context sets are fundamental for the correctness of our method. However, they cannot help us to disprove the non observationally valid clauses. For this purpose, we introduce a new notion of test context sets and we use test sets defined in [Bou97].

**Definition 5.7 (test set, test substitution)** *A* test set *is a cover set which has the following additional properties: (i) the instance of a ground reducible term by a test substitution matches a left-hand side of $R$. (ii) if the instance of a positive clause $C_{pos}$ by a test substitution $\sigma$ is strongly irreducible, then $C_{pos}\sigma$ is not inductively valid w.r.t. $R$. A* test substitution *for a clause $C$ instanciates all induction variables of $C$ by terms taken from a given test set whose variables are renamed.*

**Definition 5.8 (test context set, test clausal context)** *A* test context set *$S$ is a cover context set such that for each positive clause $C_{pos}$, if $c[C_{pos}]\sigma$ is strongly irreducible where $\sigma$ is a test substitution of $C_{pos}$ and $c$ is a test clausal context of $C_{pos}$, then $C_{pos}\sigma$ is not observationally valid w.r.t. $R$. A* test clausal context *for a clause $C$ is a clausal context for $C$ whose contexts belongs to $S$.*

Test substitutions and test contexts sets permit us to refute false conjectures by constructing a counterexample.

**Definition 5.9 (provably inconsistent)** *Let $R$ be a conditional rewriting system with observable preconditions. We say that $C$ is* provably inconsistent *if and only if there exists a test substitution $\sigma$ and a clausal test context $c$ such that:*

*(i) for all $i$, $a_i\sigma = b_i\sigma$ is an inductive theorem w.r.t. $R$.*

*(ii) $c[C_P]\sigma$ is strongly irreducible by $R$ where $C_P \equiv a'_1 = b'_1 \vee \cdots \vee a'_m = b'_m$.*

Provably inconsistent clauses are not observationally valid.

**Theorem 5.1** *Let $R$ be a conditional rewriting system with observable preconditions. Let $C$ be a provably inconsistent clause. Then $C$ is not observationally valid.*

**Proof:** Let $C$ be a provably inconsistent clause. Then, there exists a test substitution $\sigma$ and a test clausal context $c$ such that:

(i) for all $i$, $a_i\sigma = b_i\sigma$ is an inductive theorem $R$.

(ii) $c[C_P]\sigma$ is strongly irreducible w.r.t. $R$ where $C_P \equiv a'_1 = b'_1 \vee \cdots \vee a'_m = b'_m$.

$c[C_P]\sigma$ is strongly irreducible w.r.t $R$. Then, by definition of a test clausal context, $R \not\models_{obs} C_P\sigma$. Then there exists a ground substitution $\tau$, $R \not\models_{obs} C_P\sigma\tau$. On the other hand, for all $i$, $a_i\sigma = b_i\sigma$ is an inductive theorem in $R$. In particular, for all $i$, $a_i\sigma\tau = b_i\sigma\tau$ is valid in $R$. We deduce that $R \not\models_{obs} C\sigma$. □

## 5.1 Computation of test sets

The computation of test sets and test substitutions for conditional specifications is decidable if the axioms are sufficiently complete and the constructors are specified by a set of unconditional equations (see [Kou92]). Unfortunately, no algorithm exists for the general case of conditional specifications. However, in [Bou97], a procedure is described for computing test sets when the axioms are sufficiently complete over an arbitrary specification of constructors.

## 5.2 Computation of test contexts

Let us first introduce the following lemma which gives us a useful characterization of test context sets:

**Lemma 5.1** *Let $R$ be a conditional rewriting system with observable preconditions. Let $CC$ be a cover context set such that for each term $t$ of sort $s$, if $c[t]\sigma$ is strongly irreducible where $\sigma$ is a test substitution of $t$ and $c[z_s] \in CC$, there exists a ground observable context $c_{obs}$ such that $c_{obs}[t]\sigma$ is strongly irreducible. Then, $CC$ is a test context set.*

**Proof:** Let $C \equiv a_1 = b_1 \vee \ldots \vee a_n = b_n$. Suppose that $c[C]\sigma$ is strongly irreducible, where $\sigma$ is a test substitution of $C$ and $c$ is a test clausal context of $C$. Let us prove that $C\sigma$ is not observationally valid. By hypothesis, there exists $c' \equiv< c'_1, \ldots, c'_n >$, an observable test clausal context of $C$ such that $c'[C]\sigma$ is strongly irreducible. By the test contexts set property of $CC$, there exists an observable ground clausal context $c_{obs} =< c_{obs_1}, \ldots, c_{obs_n} >$, such that $c_{obs}[C]\sigma$ is strongly irreducible. Then, $c_{obs}[C]\sigma$ is a provably inconsistent clause w.r.t. Definition 11 in [Bou97]. By Theorem 12 in [Bou97], $c_{obs}[C]\sigma$ is not inductively valid. Thus $R \not\models_{obs} C\sigma$. □

Now, let us present a method of constructing such test contexts.

**Definition 5.10** *The sets $T_{obs}$ and $T_{\neg obs}$ are defined as follows:*

$T_{obs} = \{c \in T(F,X) \mid |c| \leq depth(R),\ c \in \mathcal{C}_{obs}\ ,$ *$c$ is irreducible and does not contain any observable strict subcontext* $\}$

$T_{\neg obs} = \{c \in T(F,X) \mid |c| = depth(R),\ c \notin \mathcal{C}_{obs}\ ,$ *$c$ is irreducible, $c$ does not contain any observable subcontext, and all variables (including the context one) in $c$ occur at $depth(R)$}.*

**Example 5.2** *Consider again the Stack specification in figure 1. We have $depth(R) = 1$, then: $T_{obs} = \{z_{nat}, top(z_{stack})\}$, $T_{\neg obs} = \{pop(z_{stack}), push(i, z_{stack})\}$.*

We need the following definition of the *top* of a term.

**Definition 5.11** *The top of a term is defined by:*

$$top(t,d) = t,\ \text{if } |t| \leq d$$
$$top(f(t_1,\ldots,t_n),0) = f(x_1,\ldots,x_n),\ \text{where } x_i (i \in [1..n]) \text{ are fresh variables}$$
$$top(f(t_1,\ldots,t_n),d) = f(top(t_1,d-1),\ldots,top(t_n,d-1)),\ \text{otherwise}$$

From now, we assume that $R$ is a conditional rewriting system with **observable preconditions**, **sufficiently complete over constructors**, and that **relations between constructors are left-linear and equational**. The next theorem shows that under these hypotheses it is possible to construct a test context set. Note also that with these assumptions a term $t \in T(\mathcal{C}, \mathcal{X})$ is strongly irreducible iff it is irreducible.

**Theorem 5.2** *The set* $T = \{c \in T_{\neg obs} \cup T_{obs} \mid \ c \notin T(\mathcal{C}, \mathcal{X})$ *and there exists an observable context* $c_{obs}$ *such that* $c_{obs}[c]$ *is irreducible* $\} \cup \{c \in T_{\neg obs} \cup T_{obs} \mid c \in T(\mathcal{C}, \mathcal{X})$*, and there exists an observable ground context* $c_{obs}$ *such that* $c_{obs}[c]$ *is not quasi ground reducible*$\}$ *is a test context set w.r.t. lemma 5.1.*

**Proof:** $T$ is cover context set: let $c[z_s]$ be a ground observable and irreducible context such that $c[z_s]$ does not contain any observable strict subcontext. Let $c'[z_s] = top(c[z_s], d)$ where $d$ is the length of the occurrence of $z_s$. If $|c'[z_s]| \leq depth(R)$, then $c'[z_s] \in T$, by construction of $T$. Otherwise, $|c'[z_s]| > depth(R)$. Let $c''$ be a subcontext of $c'$ such that $|c''| = depth(R)$. Then, $c'' \in T$, by construction of $T$. So, in each case, there exists a context $c'' \in T$ such that $c$ is a subcontext of $c$. We deduce that there exists a ground substitution such that $dom(\theta) = var(c)$ and $c\theta$ is a subcontext of $c$.

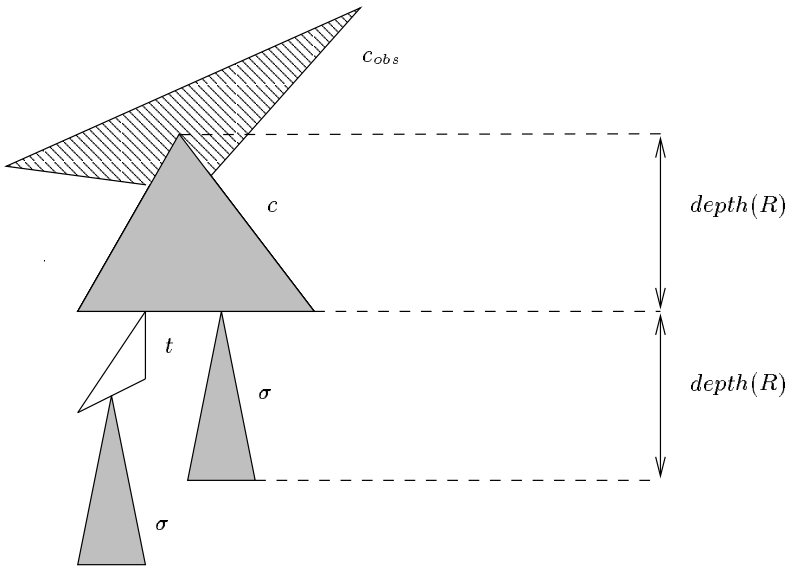
Let us check the second property: Let $t$ be a term of sort $s$, $c[z_s] \in T$ and $\sigma$ a test substitution such that $c[t]\sigma$ is strongly irreducible. Let us show that there exists a ground observable context $c_{obs}$ such that $c_{obs}[c[t]]\sigma$ is strongly irreducible. In this case, we will deduce a ground substitution $\tau$ such that $c_{obs}[c]\sigma\tau$ is a ground context and strongly irreducible (cf. test set property). Let $c'_{obs} = c_{obs}[c]\tau$. Thus, $c'_{obs}[t]\sigma$ is strongly irreducible.

Since $c[t]\sigma$ is strongly irreducible, there exists a ground substitution $\tau$ such that $c[t]\sigma\tau$ is ground and irreducible (cf property of test sets). Since $R$ is sufficiently complete over constructors, we have $c \in T(\mathcal{C}, X)$. By construction of $T$, there exists an observable ground context $c_{obs}$ such that $c_{obs}[c]$ is not quasi ground reducible:

- if $c \in \mathcal{C}_{obs}$ , then we set $c_{obs} = z_s$, where $s$ is the sort of $c$.
- if $c \notin \mathcal{C}_{obs}$ , then $|c| = depth(R)$. Let us show that $c_{obs}[c[t]]\sigma$ is strongly irreducible. Suppose that $c_{obs}[c[t]]\sigma$ is not strongly irreducible. Since the relations between constructors are equational, Let $g \to d$ a rule of $R$, $\theta$ a substitution and $u$ a position such that $c_{obs}[c[t]]\sigma/u = g\theta$. Then, the position $u \in Pos(c_{obs})$ because $c[t]\sigma$ is irreducible. Let $v$ be a strict position in $g$, then $v$ appears in $c_{obs}[c]/u$, otherwise $|v| > depth(R)$ since all variables of $c$ appear at $depth(R)$, absurd. For each variable $x$ appearing at a position $w$ in $g$, we define $x\theta' = c_{obs}[c]/uw$. Thus, $g\theta' = c_{obs}[c]/u$, which contradicts the fact that $c_{obs}[c]$ is not quasi ground reducible. Thus $c_{obs}[c[t]]\sigma$ is strongly irreducible.

**input:** $T$
$T_0 := \{c_{obs} \in (T \cap \mathcal{C}_{obs}) \mid c_{obs}$ is not quasi ground reducible $\}$.
**repeat**
  $K_i := T \setminus T_i$
  $T_{i+1} := T_i \cup \{c \in K_i \mid \exists c_i \in T_i$ such that $c_i[c]$ is not quasi ground reducible $\}$
**until** $T_{i+1} = T_i$
$T^* = T_i$
**output:** $T^*$

Figure 2: Closure procedure




□

To test whether for a given context $c \in T(\mathcal{C}, \mathcal{X})$, there exists an observable context $c_{obs}$ such that $c_{obs}[c]$ is not quasi ground reducible, we use the procedure *closure* defined in figure 2. The idea of *closure* is inspired from the one used in [BK89] for the ground reducibility test for an equational rewriting system: starting from the non quasi ground reducible observable contexts of depth smaller than $depth(R)$, we construct all contexts that can be embedded in one of those observable contexts, to give a non quasi ground reducible and observable context. The procedure *Closure* terminates for each finite input $T$, since $T^* \subseteq T$.

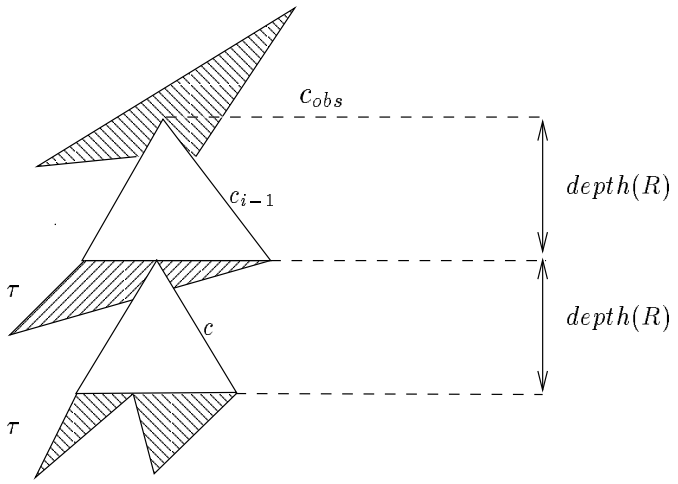
**Theorem 5.3** *Let $T = T_{obs} \cup T_{\neg obs}$. Let $T^*$ be the Closure of $T$. We have:*

- **completeness:** *For all $c \in T \setminus T^*$, for all $c_{obs} \in \mathcal{C}_{obs}$, $c_{obs}[c]$ is quasi ground reducible.*

- **minimality:** *For all $c \in T^*$, there exists a ground context $c_{obs} \in \mathcal{C}_{obs}$ such that $c_{obs}[c]$ is not ground reducible.*

**Proof:**

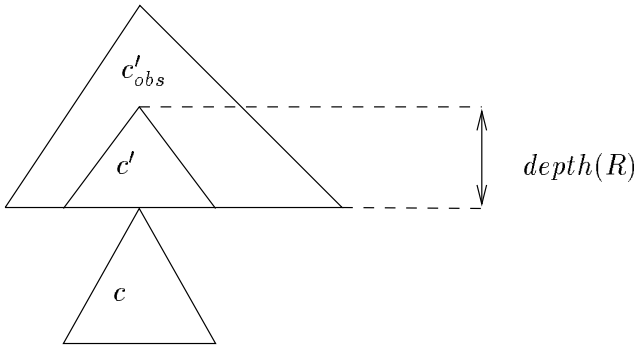
- minimality:
  Let $c \in T^*$. Then, there exists $i$ such that $c \in T_i$. Let us show $\forall i \in \mathbb{N}$, there exists a ground observable context such that $c_{obs}[c]$ is not quasi ground reducible. The proof is by induction on $i$.
  - case where $i = 0$: in this case, $c \in T_0$. Then, we set $c_{obs} = z_s$, where $s$ is the sort of $c$.
  - case where $i > 0$: there exists $c_{i-1} \in T_{i-1}$ such that $c_{i-1}[c]$ is not quasi ground reducible. If $c_{i-1} \in \mathcal{C}_{obs}$ , then we set $c_{obs} = c_{i-1}\tau$, where $\tau$ is a ground substitution such that $dom(\tau) = var(c_{i-1})$ and $c_{i-1}\tau$ is irreducible. Otherwise, $|c_{i-1}| = depth(R)$. By induction hypothesis, there exists a ground observable context $c_{obs}$ such that $c_{obs}[c_{i-1}]$ is not quasi ground reducible. Let us show that $c_{obs}[c_{i-1}[c]]$ is not quasi ground reducible. Since $c_{i-1}[c]$ is not ground reducible, there exists a ground substitution $\tau$ such that $dom(\tau) = var(c) \cup var(c_{i-1})$ and $c_{i-1}[c]\tau$ is irreducible. Since $|c_{i-1}| = depth(R)$ ($c_{i-1} \notin \mathcal{C}_{obs}$ ), $R$ is left-linear and $c_{i-1}[c]\tau$ is irreducible, we can show that $c_{obs}[c_{i-1}[c]]\tau$ is also irreducible, as in Theorem 5.2. Thus, $c_{obs}[c_{i-1}[c]]$ is not quasi ground reducible.




- completeness:
  Suppose there exists $c \in T \setminus T^*$ such that there exists an observable ground context $c_{obs}$ and $c_{obs}[c]$ is not quasi ground reducible. Let $d$ be the length of the occurrence of $z_s$ in $c_{obs}$ . Let $c'_{obs} = top(c_{obs}, d)$. Then $c'_{obs}[c]$ is not quasi ground reducible. Let us

choose $c$ such that $|c'_{obs}[c]|$ is minimal.
Note that $c'_{obs} \notin T^*$, otherwise $c \in T^*$.

- if $|c'_{obs}| \leq depth(R)$, then $c'_{obs} \in T_0 \subseteq T^*$ which contradicts that $c'_{obs} \notin T^*$.
- if $|c'_{obs}| > depth(R)$, then let $c'$ be the subcontext of $c'_{obs}$ with the same context variable, such that $|c'| = depth(R)$. If $c' \in T^*$, then $c \in T^*$, which contradicts the hypothesis. Thus, $c' \in T \setminus T^*$ and $|c'_{obs}[c']| < |c'_{obs}[c]|$, which contradicts that we have chosen a context $c \in T \setminus T^*$ such that $|c'_{obs}[c]|$ is minimal.




□

The quasi ground reducibility is decidable since the relations between constructors are equational, as it is shown by the next lemma:

**Lemma 5.2** *The quasi ground reducibility is decidable for equational rewriting systems.*

**Proof:** The ground reducibility is decidable for equational systems [KNZ87]. Let $c[z_s]$ be a context. Let $a$ be a function symbol such that $a \notin F$, and $arity(a) = 0$. We set $t = c[a]$. Let us show that:

$$(\forall \tau \text{ ground on } T(F),\ t\tau \text{ reducible}) \Leftrightarrow (c[z_s] \text{ quasi ground reducible})$$

We will then deduce the decidability of ground reducibility.

- $\Rightarrow$:
  Suppose $t\tau$ reducible. Then, there exists a substitution $\theta$, a position $u$ in $t\tau$ and a rule $g \to d$ in $R$ such that $t\tau/u = g\theta$. Let $v$ be the position of $a$ in $t$. Since $a \notin F$, there necessarily exists a variable $y$ in $g$ at a position $v' \preceq v$, such that $y\theta = c'[a]$, where $c'$ is a subcontext of $c$. Then, we define a substitution $\theta'$ such that $x\theta' = c'[z_s]$ if $x = y$ and $x\theta' = x\theta$ otherwise. Then, $c[z_s]\tau'/u = g\theta'$, where $\tau'$ is the restriction of $\tau$ to $var(c)$.

- $\Leftarrow$:
  Suppose $c[z_s]$ quasi ground reducible. Let $\tau$ a ground substitution such that $dom(\tau) = var(c)$. Then, $c[z_s]\tau$ is reducible. In particular, $c[a]\tau$ is reducible.

□

Thanks to Lemma 5.2, we can easily refine the construction of test contexts in the case of a left-linear equational rewriting system.

**Theorem 5.4** *Let $R$ a left-linear equational rewriting system. The set $TC = \{c \in (T_{obs} \cup T_{\neg obs}) \mid$ there exists an observable ground context $c_{obs}$ such that $c_{obs}[c]$ is not quasi ground reducible$\}$ is a test context set according to Definition 5.8.*

**Example 5.3** *Consider the sets $T_{obs}$ and $T_{\neg obs}$ of example 1. Applying the procedure* closure*, we obtain:*
$T_0 = \{z_{nat}, top(z_{stack})\}$
$K_0 = \{pop(z_{stack}), push(i, z_{stack})\}$
$T_1 = T_0 \cup \{pop(z_{stack})\}$
$K_1 = \{push(i, z_{stack})\}$
$T_2 = T_1$.
*Thus $T^* = T_1 = \{z_{nat}, top(z_{stack}), pop(z_{stack})\}$, and $T_1$ is a test context set for the specification stack.*

**Example 5.4** *Consider the List specification in figure 3. We have: $depth(R) = 1$, $T_{obs} = \{z_{nat}, z_{bool}, in(x, z_{list})\}, T_{\neg obs} = \{z_{nat}, z_{bool}, union(z_{list}, x), union(x, z_{list}), insert(x, z_{list})\}$.*
*Applying the closure algorithm, we get the test context set:*

$$T = \{z_{nat}, z_{bool}, in(x, z_{list}), union(z_{list}, x), union(x, z_{list})\}$$

*In fact: $in(y, union(z_{list}, x))$ and $in(y, union(x, z_{list}))$ are irreducible contexts, so $union(z_{list}, x)$ and $union(x, z_{list})$ are in $T$. But $insert(x, z_{list})$ is not in $T$, since $in(y, insert(x, z_{list}))$, $union(y, insert(x, z_{list}))$ and $union(insert(x, z_{list}), y)$ are not quasi ground reducible (union and in are defined operators).*

It is possible to reduce test contexts sets in the case where the rewriting system is sufficiently complete over constructors. In fact, we can just consider a subset $T'$ of $T$, such that $T' = \{c[z_s] \in T \mid$ if $c \notin T(\mathcal{C}, \mathcal{X})$, then $z_s$ appears in an induction position of a completely defined operator $\}$. Intuitively, the induction variables are those which may trigger a rewriting step when instanciated. Then, if a contextual variable $z_s$ of a context $c[z_s]$ appears in an induction position of a completely defined operator, the application of $c$ to a term $t$ $(\in T(\mathcal{C}, \mathcal{X}))$ of sort $s$ is unimportant, since the result is independent from $t$.

**Example 5.5** *Consider ageain example 3. The contextual variable of the contexte $union(x, z_{list})$ appears in an induction position d'induction of union. Then, we consider the test contexts set $\{z_{nat}, z_{bool}, in(x, z_{list}), union(z_{list}, x)\}$.*

**specification:** LIST
**sorts:** nat, bool, list
**observable sorts:** nat, bool
**constructors:**
$0$: $\rightarrow$nat;
$s$: nat $\rightarrow$nat;
$Nil$: $\rightarrow$list;
$insert$: nat $\times$ list $\rightarrow$list;
$True$: $\rightarrow$bool;
$False$: $\rightarrow$bool;
**defined functions:**
$union$: list $\times$ list $\rightarrow$ list;
$in$: nat $\times$ list $\rightarrow$ bool;
$eq$: nat $\times$ nat $\rightarrow$ bool;
**axioms:**
$union(Nil, l) = l$
$union(insert(x, l), l1) = insert(x, union(l, l1))$
$in(x, Nil) = False$
$eq(x, y) = True => in(x, insert(y, l)) = True$
$eq(x, y) = False => in(x, insert(y, l)) = in(x, l)$
$eq(x, x) = True$
$eq(0, s(x)) = False$
$eq(s(x), 0) = False$
$eq(s(x), s(y)) = eq(x, y)$

Figure 3: Specification List

# 6 Inference system

The inference system we use (see figure 4) is based on a set of transition rules applied to $(E, H)$, where $E$ is the set of conjectures to prove and $H$ is the set of induction hypotheses. The initial set of conditional rules $R$ is oriented with a well founded ordering. An *I-derivation* is a sequence of states: $(E_0, \emptyset) \vdash_I (E_1, H_1) \vdash_I \ldots (E_n, H_n) \vdash_I \ldots$. We say that an I-derivation is *fair* if the set of persistent clauses $(\cup_i \cap_{j \geq i} E_j)$ is empty. Context induction is performed implicitly by the **Generation** rule. An equation is selected in a clause and it is extended by test contexts. These extensions are rewritten by $R$ either by case analysis or by standard conditional rewriting. The resulting conjectures are collected in $\bigcup E_{c,\sigma}$. Case Analysis illustrates the case reasoning: it simplifies a conjecture with conditional rules provided that the disjunction of their preconditions is inductively valid in $R$.

**Definition 6.1 (Case Analysis)** *Let $R$ be a set of conditional rules and let $C$ be a clause.* $\text{CaseAnalysis}(C[g\sigma]_u) = \{P_1\sigma \Rightarrow C[d_1\sigma]_u; \cdots; P_n\sigma \Rightarrow C[d_n\sigma]_u\}$ **if** $\forall i \in [1 \cdots n] : P_i \Rightarrow g \rightarrow d_i \in R$ **and** $R \models_{ind} P_1\sigma \vee \cdots \vee P_n\sigma$.

**Generation**: $(E \cup \{\underbrace{\vee_{i=1}^{n} e_i}_{C}\}, H) \vdash_I (E \cup (\bigcup E_{c,\sigma}), H \cup \{C\})$

if for all test substitution $\sigma$ of $C$, there exists $k$ such that for all test context $c$:
either $(c[e_k]\sigma) \vee C_{rest}$ is a tautology, then $E_{c,\sigma} = \emptyset$
either $c[e_k]\sigma \rightarrow_R e'_k$, then $E_{c,\sigma} = \{e'_k \vee C_{rest}\}$
either $E_k = \mathit{CaseAnalysis}(c[e_k]\sigma)$, then $E_{c,\sigma} = \bigcup_{(P' \Rightarrow e') \in E_k} \{P' \Rightarrow e' \vee C_{rest}\}$
where $C_{rest} \equiv (\bigvee_{i \in [1..k-1,k+1,..n]}(e_i\sigma))$

**Case Simplification**: $(E \cup \{C\}, H) \vdash_I (E \cup E', H)$

if $E' = \mathit{Case\ Analysis}(C)$

**Simplification**: $(E \cup \{(a = b)^\epsilon \vee r\}, H) \vdash_I (E \cup \{(a' = b)^\epsilon \vee r\}, H)$

*if* $a \rightarrow_R a'$
*or* $a[v\lambda] \rightarrow_{H \cup E} a[w\lambda]$ by $v = w$ where $v \succ w$ and ($v\lambda \prec a$ or $w\lambda \prec b$)

**Subsumption**: $(E \cup \{C\}, H) \vdash_I (E, H)$

if $C$ is subsumed by another clause of $(E \setminus \{C\}) \cup H \cup R$

**Delete**: $(E \cup \{C\}, H) \vdash_I (E, H)$

if $C$ is a tautology

**Disproof**: $(E \cup \{C\}, H) \vdash_I \mathit{Disproof}$

if $C$ is provably inconsistent

Figure 4: Inference System $I$

An I-derivation *fails* when there exists a conjecture such that no rule can be applied to it. An I-derivation *succeeds* if all conjectures are proved. A clause $C$ will be now denoted as the disjunction of its literals. Let $(a = b)^\epsilon$ be the literal $(a = b)$ (resp., $\neg(a = b)$) if $\epsilon = +$ (resp., $\epsilon = -$) . The application of a context $c$ to a literal $(a = b)^\epsilon$, denoted by $c[(a = b)^\epsilon]$, is $(c[a] = c[b])^\epsilon$. We write $(a = b \rightarrow_R a' = b)$ or $(b = a \rightarrow_R a' = b)$, if $(a \rightarrow_R a')$ and $(a \succ b$ or $a \not\succ b)$.

## 6.1 Correctness

Correctness is proved by considering the minimal counterexample w.r.t. the following ordering, which is inspired from [Lys94]:

**Definition 6.2** *The well-founded ordering on couples of clauses and clausal context is defined by first introducing the complexity of a couple (equation, context). The complexity of a couple $(g = h, c)$ is defined as:*

$$\mathcal{C}((g = h, c)) \;=\; \begin{cases} (\{c[g]\}, \{g\}, \{c[h]\}) & if \quad g \succ h \\ (\{c[h]\}, \{h\}, \{c[g]\}) & if \quad h \succ g \\ (\{c[g], c[h]\}, \bot, \bot) & otherwise \end{cases}$$

*where the new symbol $\bot$ is taken to be minimal in $\ll$. We define an ordering on equations as follows: $(a = b, c) \prec_e (a' = b', c')$ iff $\mathcal{C}((a = b, c))$ is smaller than $\mathcal{C}((a' = b', c'))$ for the lexicographic composition of $\ll$ on the first and second components of the complexity. The multiset extension of $\prec_e$ will be denoted by $\ll_e$.*

*Let $C$ be a clause of type $\bigwedge_{i\in[1..m]} a_i = b_i \Rightarrow \bigvee_{j\in[1..n]} h_j = k_j$ and a clausal context $c =< c_1, \ldots, c_n >$. We define $Rep((C, c)) = \{\mathcal{C}((a_i = b_i, x_{s_i}))\}_{i\in[1..m]} \cup \{\mathcal{C}((h_j = k_j, c_j))\}_{j\in[1..n]}$, where for each $i \in [1..m]$ $s_i$ is the sort of $a_i$. Given two clauses $C_1$, $C_2$, with respective clausal contexts $c_1$, $c_2$ we say that $(C_1, c_1) \prec_{\mathcal{C}} (C_2, c_2)$ if lexicographically $Rep((C_1, c_1)) \ll_e Rep((C_2, c_2))$ or $nln(C_1) < nln(C_2)$, where $nln(C)$ is the number of negative literals of $C$.*

**Theorem 6.1 (correctness of successful I-derivations)** *Let $(E_0, \emptyset) \vdash_I (E_1, H_1) \vdash_I \ldots$ be a fair I-derivation. If it does not fail then $R \models_{obs} E_0$.*

**Proof:** Suppose $R \not\models_{obs} E_0$ and let $(C', c')$ be a minimal element w.r.t. $\prec_{\mathcal{C}}$ of the set $\{(C\theta, c_{obs}) \mid R \not\models c_{obs}[C]\theta$ where $C \in \cup_i E_i$, $\theta$ is a ground irreducible substitution and $c_{obs}$ is a ground observable clausal context of $C\}$. $C'$ exists since $R \not\models_{obs} E_0$ and $\prec_{\mathcal{C}}$ is well founded. Then, there exist a clause $C \equiv (\vee_{i=1}^{n} e_i) \in \cup_i E_i$, minimum w.r.t subsumption ordering, an irreducible ground substitution $\theta$ and a ground observable clausal context $c_{obs} =< c^1_{obs}, \ldots, c^n_{obs} >$ such that $c'[C'] = c_{obs}[C]\theta$. Now, we show that no inference rule can be applied to $C$. This shows that the derivation fails since $C$ must not persist in the derivation by the fairness hypothesis. Hence let us assume that $C \in E_j$ and $(E_j, H_j) \vdash_I (E_{j+1}, H_{j+1})$ by some rule applied to $C$. We discuss now the situation according to which rule is applied. In every case we shall derive a contradiction. In order to simplify the notations we write $E$ for $E_j$ and $H$ for $H_j$.

**Generation:** Suppose that the rule *Generation* is applied to $C$. Since $c_{obs}[C]\theta$ is ground and irreducible, there exist a test clausal context $c =< c_1, \ldots, c_n >$, a ground observable clausal context $c_o =< c^1_o, \ldots, c^n_o >$ a ground substitution $\tau$ and a test substitution $\sigma$ such that $c_{obs} = c_o[c\theta]$, $\sigma\tau = \theta$ and $c_{obs}[C]\theta = c_o[c[C]\theta]$.

a) if there exists $i$ such that $c_i[e_i]\sigma \vee (\bigvee_{j\in[1..k-1,k+1,..n]}(e_j\sigma))$ is a tautology, then $R \models c_{obs}[C]\theta$, absurd.

b) suppose that there exists $i$ such that $e_i \equiv (a_i = b_i)^{\epsilon_i}$ and $c_i[a_i]\sigma \rightarrow_R a'_i$. Let $e'_i \equiv (a'_i = b_i)^{\epsilon_i}$ and let $C' \equiv e'_i \vee (\bigvee_{j\in[1..i-1,i+1,..n]}(c_j[e_j]\sigma))$. Then $R \not\models c_o[C']\tau$. On the other hand, $a'_i \prec c_i[a_i]\sigma$, then $a'_i\tau \prec c_i[a_i]\sigma\tau$, because $\prec$ is stable per instanciation,

and $c_o[a'_i]\tau \prec c^i_{obs}[a_i]\sigma\tau$, because $\prec$ is stable per context. Thus $(C'\tau, c_o) \prec_{\mathcal{C}} (C\theta, c_{obs})$, absurd.

c) if *Case Analysis* is applied to $c_i[e_i]\sigma$, then there exist $P_1 \Rightarrow g = d_1, \ldots, P_n \Rightarrow g = d_n \in R$ such that $c[e_i]\sigma/u = g\phi$ and $R \models_{obs} P_1\phi \vee \ldots \vee P_m\phi$.

The result of application of *Case Analysis* gives the clauses

$$\{C_k \equiv P_k\phi \Rightarrow (c_i[e_i]\sigma)[d_k\phi]_u \vee (\bigvee_{j\in[1..i-1,i+1,..n]} e_j\sigma)\}_{k\in[1..m]}$$

Since $R \models_{obs} P_1\phi \vee \ldots \vee P_n\phi$, there exists $k$ and a ground substitution $\tau$ such that $R \models P_k\phi\tau$ and therefore $R \models g\phi\tau = d_k\phi\tau$.

Let $C' \equiv P_k\phi \Rightarrow (c_i[e_i]\sigma)[d_k\phi]_u \vee (\bigvee_{j\in[1..i-1,i+1,..n]} c_j[e_j\sigma])$. Then, $R \not\models c_o[C'\tau]$. On the other hand, $(C'\tau, c_o) \prec_{\mathcal{C}} (C\theta, c_{obs})$ since $\{P_k\phi\tau, d_k\phi\tau\} \ll \{g\phi\tau\}$, absurd.

**Case Simplification:** This case is similar to the previous one.

**Subsumption**: Since $R \not\models c_{obs}[C]\theta$, $C$ cannot be subsumed by an axiom of $R$. If there exists $C' \in H \cup (E \setminus \{C\})$ such that $C = C'\rho \vee r$, then $R \not\models c_{obs}[C']\rho\theta$ and therefore $r = \emptyset$ and $\rho = \mathcal{I}$ since $C$ is minimal in $\cup_i E_i$ w.r.t subsumption ordering. Thus, $C' \notin (E \setminus \{C\})$. On the other hand, $C' \notin H$, otherwise, it means that *generation* was applied to $C$, absurd.

**Simplification**: Assume that **Simplification** is applied to $C \equiv (a_i = b_i)^{\epsilon_i} \vee r$, there are two cases:

a) if $a_i \rightarrow_R a'$, then $R \not\models (c_{obs}[a' = b_i])^{\epsilon_i}\theta \vee c_{obs}[r]\theta$. On the other hand, $(((a' = b_i)^{\epsilon_i} \vee r)\theta, c_{obs}) \prec_{\mathcal{C}} (C\theta, c_{obs})$ since $a_i \succ a'$, absurd.

b) if $a_i[v\lambda] \rightarrow_{H\cup E} a' \equiv a_i[w\lambda]$ by $v = w$ where $v \succ w$ and ( $v\lambda \prec a_i$ or $w\lambda \prec b_i$). Then $(((a' = b_i)^{\epsilon_i} \vee r)\theta, c^i_{obs}) \prec_{\mathcal{C}} (C\theta, c_{obs})$ and therefore $R \models (c^i_{obs}[a' = b_i]\theta)^{\epsilon_i}$. So $R \not\models (c_{obs}[a' = b_i]\theta)^{\epsilon_i} \vee c_{obs}[r]\theta$. On the other hand, $(((a' = b_i)^{\epsilon_i} \vee r)\theta, c_{obs}) \prec_{\mathcal{C}} (C\theta, c_{obs})$ since $v \succ w$, absurd.

□

**Theorem 6.2 (correctness of disproof)** *Let* $(E_0, \emptyset) \vdash_I (E_1, H_1) \vdash_I \cdots$ *be an I-derivation. If there exists* $j$ *such that* **Disproof** *is applied to* $(E_j, H_j)$*, then* $R \not\models_{obs} E_0$.

**Proof:** If there exists $j$ such that **Disproof** is applied to $(E_j, H_j)$, then by Theorem 5.1, we conclude that $R \not\models_{obs} E_j$. Now, to prove that $R \not\models_{obs} E_0$, it is sufficient to prove the following claim: Let $(E_j, H_j) \vdash_I (E_{j+1}, H_{j+1})$ be an I-derivation step. If $\forall i \leq j, R \models_{obs} E_i$ then $R \models_{obs} E_{j+1}$. The proof is by simple case analysis on the rules applied on a clause

$C \equiv \vee_{j\in[1..n]}e_j \quad \in E_i$.

**Generation:** If **Generation** is applied on $C$, then for all test substitution $\sigma$, there exists $i$ such that $e_i \equiv (a_i = b_i)^{\epsilon_i}$ and for all test context $c_i$:

a) either $(c_i[e_i]\sigma) \vee (\bigvee_{j\in[1..k-1,k+1,..n]}(e_j\sigma))$ is a tautology, in this case $E_{c_i,\sigma} = \emptyset$.

b) either $c_i[a_i]\sigma \rightarrow_R a'_i$. Let $e'_i \equiv (a'_i = b_i)^{\epsilon_i}$. All rules used in the rewriting step are valid. Thus $R \models_{obs} (e'_i) \vee (\bigvee_{j\in[1..i-1,i+1,..n]}(e_j\sigma))$.

c) either *Case Analysis* is applied on $c_i[e_i]\sigma$. Then there exist $P_1 \Rightarrow g = d_1, \ldots, P_n \Rightarrow g = d_n \in R$ such that $c[e_i]\sigma/u = g\phi$ and $R \models_{obs} P_1\phi \vee \ldots \vee P_m\phi$.

The result of application of *Case Analysis* gives the clauses

$$\{C_k \equiv P_k\phi \Rightarrow ((c_i[e_i]\sigma)[d_k\phi]_u) \vee (\bigvee_{j\in[1..i-1,i+1,..n]}(e_j\sigma))\}_{k\in[1..m]}$$

Suppose that there exists $k$ such that $R \not\models_{obs} C_k$. Then, there exists $\tau$ ground such that $R \not\models C_k\tau$. Since the preconditions are observable, we have $R \models P_k\phi\tau$ but $R \not\models_{obs} ((c_i[e_i]\sigma\tau)[d_k\phi\tau]_u) \vee (\bigvee_{j\in[1..i-1,i+1,..n]}(e_j\sigma\tau))\}_{k\in[1..m]})$. But since $R \models P_k\phi\tau$, we have $R \models_{obs} g_k\phi\tau = d_k\phi\tau$. Since $R \models_{obs} C$, we have $R \models_{obs} C_k\tau$, absurd.

**Case Simplification:** This case is similar to the previous one.

**Subsumption and Delete:** If $C$ is a tautology or subsumed by a clause of $R \cup E_i$, then $C$ is deleted. So $E_{j+1} \subset E_j$, and $R \models_{obs} E_{j+1}$.

**Simplification:** If the rule **Simplification** is applied on $C$, then $C \rightarrow_R C'$. All the instances of clauses used in the rewriting step occur in some $E_i$, $i \leq j$, so they are observationally valid. Thus, $R \models_{obs} C'$.

□

It is interesting to improve the efficiency of our inference system $I$ by adding the rules of the inference system $J$ (see Figure 5). Let $I'$ be the new inference system obtained. The rule *Context subsumption* appears to be useful particularly for manipulating non orientable conjectures. Based on the techniques above, we prove that the inference system including $J$ remains correct and refutationally correct.

**Theorem 6.3 (correctness of successful J-derivations)** *Let $R$ be a ground convergent rewriting system. Let $(E_0, \emptyset) \vdash_J (E_1, H_1) \vdash_J \ldots$ a fair J-derivation. If it does not fail, then $R \models_{obs} E_0$.*

**Proof:** Suppose $R \not\models_{obs} E_0$ and let $(C', c')$ be a minimal element w.r.t. $\prec_{\mathcal{C}}$ of the set $\{(C\theta, c_{obs}) \mid R \not\models c_{obs}[C]\theta$ where $C \in \cup_i E_i$, $\theta$ is a ground irreducible substitution and $c_{obs}$ is

a ground observable clausal context of $C\}$. $C'$ exists since $R \not\models_{obs} E_0$ and $\prec_{\mathcal{C}}$ is well founded. Then, there exist a clause $C \equiv \vee_{i=1}^{n} e_i^{\epsilon_i}$, minimum w.r.t subsumption ordering, an irreducible ground substitution $\theta$ and a ground observable clausal context $c_{obs} =< c_{obs}^1, \ldots, c_{obs}^n >$ such that $c'[C'] = c_{obs}[C]\theta$. Now, we show that no inference rule can be applied to $C$. This shows that the derivation fails since $C$ must not persist in the derivation by the fairness hypothesis. Hence let us assume that $C \in E_j$ and $(E_j, H_j) \vdash_J (E_{j+1}, H_{j+1})$ by some rule applied to $C$. We discuss now the situation according to which rule is applied. In every case we shall derive a contradiction. In order to simplify the notations we write $E$ for $E_j$ and $H$ for $H_j$.

**Positive Decomposition:** suppose that the rule **Positive Decomposition** is applied on

$$C \equiv f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n) \vee r$$

Then $R \not\models (f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n))\theta$, because $f$ has an observable sort. Since $f$ is free, there exists $i$ such that $R \not\models (s_i = t_i)\theta$. Let $C_i \equiv ((s_i = t_i) \vee r)\theta$ and $c'_{obs} =< f(s_1\theta, \ldots, s_{i-1}\theta, z_i, s_{i+1}\theta, \ldots, s_n\theta), c_{obs}^2, \ldots, c_{obs}^n >$. Since $f$ is free and has an observable sort, then $R \not\models_{obs} c'_{obs}[C_i]\theta$. Besides, $C_i \in (\cup_i E_i)$ and $(C_i, c'_{obs}) \prec_{\mathcal{C}} (C_i\theta, c_{obs})$, absurd.

**Positive Conflict:** suppose that the rule **Positive Conflict** is applied on

$$C \equiv f(s_1, \ldots, s_n) = g(t_1, \ldots, t_n) \vee r$$

let $c'_{obs} =< c_{obs}^2, \ldots, c_{obs}^n >$. Then, $R \not\models r\theta$, $r \in (\cup_i E_i)$ and $(r\theta, c'_{obs}) \prec_{\mathcal{C}} (C\theta, c_{obs})$, absurd.

**Negative Decomposition:** suppose that the rule **Negative Decomposition** is applied on

$$C \equiv \neg f(s_1, \ldots, s_m) = f(t_1, \ldots, t_m) \vee r$$

Then, $R \models (f(s_1, \ldots, s_m) = f(t_1, \ldots, t_m))\theta$, because $f$ has an observable sort. Since $R$ is ground convergent and $f$ is a free constructor, then for all $i$, $R \models (s_i = t_i)\theta$. Let $C' \equiv \vee_i \neg(s_i = t_i) \vee r$, $C' \in (\cup_i E_i)$. Let $c'^{i}_{obs} = f(s_1\theta, \ldots, s_{i-1}\theta, z_i, s_{i+1}\theta, \ldots, s_m\theta)$. We have, $R \not\models_{obs} c'^{i}_{obs}[\neg(s_i = t_i)]\theta$. Let $c'_{obs} =< c'^{1}_{obs}, \ldots, c'^{m}_{obs}, c_{obs}^2, \ldots, c_{obs}^n >$. Then, $R \not\models_{obs} c'_{obs}[C']\theta$ and $(C'\theta, c_{obs}) \prec_{\mathcal{C}} (C\theta, c'_{obs})$.

**Negative Conflict:** suppose that the rule **Negative Conflict** is applied on

$$C \equiv \neg f(s_1, \ldots, s_n) = g(t_1, \ldots, t_n) \vee r$$

Since $R$ is ground convergent on ground terms and $f$ and $g$ are distinct free constructors of observable sort, then $R \not\models (f(s_1, \ldots, s_n) = g(t_1, \ldots, t_n))\theta$. Per consequent, $R \models_{obs} C\theta$, absurd.

**Context Subsumption:** If there exists a clausal context $c$, a clause $C' \in (H \cup R \cup E \setminus \{C\})$ such that $C = c[C']\rho \vee r$. We have $C' \notin R$ otherwise $R \models_{obs} c_{obs}[C]\theta$. If $C' \in E \setminus \{C\}$, then $R \not\models_{obs} c[C']\rho$. In this case, $r = \emptyset$ and $\rho = \mathcal{I}$ otherwise $(C'\rho\theta, c_{obs}[c]) \prec_{\mathcal{C}} (C\theta, c_{obs})$, which contradicts the minimality of $C\theta$ w.r.t. $\prec_{\mathcal{C}}$. On the other side, $C' \notin H$, otherwise the rule **Generation** would be applied on $C$. □

We also show that the inference system $J$ is refutationally correct.

**Theorem 6.4 (correctness of Disproof)** *Let $R$ be a ground convergent rewriting system. Let $(E_0, \emptyset) \vdash_J (E_1, H_1) \vdash_J \cdots$ be a J-derivation. If there exists $j$ such that* **Disproof** *is applied to $(E_j, H_j)$, then $R \not\models_{obs} E_0$.*

**Proof:** If there exists $j$ such that **Disproof** is applied to $(E_j, H_j)$, then according to Theorem 5.1, we conclude that $R \not\models_{obs} E_j$. Now to prove that $R \not\models_{obs} E_0$, it is sufficient to prove the following assertion: Let $(E_j, H_j) \vdash_J (E_{j+1}, H_{j+1})$ be a J-derivation step. if $\forall i \leq j, R \models_{obs} E_i$ then $R \models_{obs} E_{j+1}$. The proof is by simple case analysis on $C \equiv \vee_{j \in [1..n]} e_j \quad \in E_i$.

**Positive Decomposition:** Suppose that the rule **Positive Decomposition** is applied to

$$C \equiv f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n) \vee r$$

Let $\theta$ be a ground substitution. Since $R \models_{obs} C\theta$, necessarily $R \models_{obs} r\theta$ or $R \models_{obs} (f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n))\theta$. Let $C_i = (s_i = t_i) \vee r$. If $R \models_{obs} r\theta$ then $R \models_{obs} C_i\theta$. Otherwise, $R \models_{obs} (f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n))\theta$, then $R \models (f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n))\theta$ because $f$ has an observable sort. Since $R$ is convergent on the ground terms and $f$ is a free constructor, for all $i$ we have: $R \models (s_i = t_i)\theta$. Thus, $R \models_{obs} C_i\theta$.

**Positive conflict:** Suppose that the rule **Positive Conflict** is applied to

$$C \equiv f(s_1, \ldots, s_n) = g(t_1, \ldots, t_n) \vee r$$

Let $\theta$ be a ground substitution. Since $R$ is ground convergent and $f$ and $g$ are free constructors with observable sort, then $R \not\models (f(s_1, \ldots, s_n) = g(t_1, \ldots, t_n))\theta$. Consequently, $R \models_{obs} r\theta$.

**Negative Decomposition:** Suppose that the rule **Negative Decomposition** is applied to

$$C \equiv \neg f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n) \vee r$$

Let $\theta$ be a ground substitution. Let $C' = \vee_i \neg(s_i = t_i) \vee r$. Since $R \models_{obs} r\theta$ then $R \models_{obs} C'\theta$. Otherwise $R \not\models_{obs} (f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n))\theta$. Then, $R \not\models (f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n))\theta$, because $f$ has an observable sort. So, there

**Positive Decomposition**: $(E \cup \{f(\vec{s}) = f(\vec{t}) \vee r\}, H) \vdash_I (E \cup (\cup_i \{s_i = t_i \vee r\}), H)$
if $f$ is a free constructor of sort in $S_{obs}$ and for all $i$: $s_i \prec t_i$

**Positive Conflict**: $(E \cup \{f(\vec{s}) = g(\vec{t}) \vee r\}, H) \vdash_I (E \cup \{r\}, H)$
if $f$ and $g$ are free distinct constructors of sort in $S_{obs}$

**Negative Decomposition**: $(E \cup \{\neg f(\vec{s}) = f(\vec{t}) \vee r\}, H) \vdash_I (E \cup \{\vee_i \neg s_i = t_i \vee r\}, H)$
if $f$ is a free constructor of sort in $S_{obs}$ and for all $i$, $s_i \prec t_i$

**Negative Conflict**: $(E \cup \{\neg f(\vec{s}) = g(\vec{t}) \vee r\}, H) \vdash_I (E, H)$
if $f$ and $g$ are free distinct constructors of sort in $S_{obs}$

**Context Subsumption**: $(E \cup \{C\}, H) \vdash_I (E, H)$
if there exists a clausal context $c$ and $C'$ in $(R \cup (E \setminus \{C\}) \cup H)$ such that $c[C']$ subsumes $C$

Figure 5: Inference system $J$

exists $i$ such that $R \not\models (s_i = t_i)\theta$. Thus, $R \models \vee(\neg(s_i = t_i))\theta$. Par consequent $R \models_{obs} C'\theta$.

**Negative conflict, Context Subsumption:** Suppose that the rule **Negative Conflict** or **Context Subsumption** is applied on $C$. Then $C$ is deleted from $E_{j+1}$. Thus, $E_{j+1} \subseteq E_j$. Per consequent $R \models_{obs} E_{j+1}$.

□

## 6.2 Refutational completeness

From now, we only consider boolean specifications. To be more specific, we assume there exists an observable sort *bool* with two free constructors $\{true,\ false\}$. The sort *bool* will be observable. Every rule in $R$ is of type: $\wedge_{i=1}^{n}\ p_i = p'_i \Rightarrow s \rightarrow t$ where for all $i$ in $[1 \cdots n]$, $p'_i \in \{true,\ false\}$. Conjectures will be *boolean clauses*, i.e. clauses whose negative literals are of type $\neg(p = p')$ where $p' \in \{true, false\}$. We denote by $\bar{p'}$, the complement of $p'$, that is $false$ if $p' = true$ and $true$ if $p' = false$. Let $f \in \mathcal{D}$, a completely defined symbol in $R$. Then $f$ is *strongly complete w.r.t* $R$ if for all the rules $p_i \Rightarrow f(t_1, \ldots, t_n) \rightarrow r_i$ whose left-hand sides are identical up to a renaming $\mu_i$, we have $R \models_{ind} \bigvee_{i=1}^{n} p_i\mu_i$. We say that $R$ is *strongly complete* if for all $f \in \mathcal{D}$, $f$ is strongly complete w.r.t $R$. We add the rule *complement* which transforms a boolean clause into a positive one, which is easier to refute:

$$\textbf{complement}: (E \cup \{\neg(a = b) \vee r\}, H) \vdash_I (E \cup \{(a = \bar{b}) \vee r\}), H)$$
$$\text{if } b \in \{true, false\}$$

**Theorem 6.5 (Refutational completeness)** *Let $R$ be a strongly complete rewrite system. Let $E_0$ be a set of boolean clauses. Then $R \not\models_{obs} E_0$ iff all fair derivations issued from $(E_0, \emptyset)$ fail.*

**Proof:**
$\Rightarrow$
By Theorems 6.1 and 6.3
$\Leftarrow$
Let us assume that $E_0$ contains only boolean clauses. The only rule that permits us to introduce negative clauses is **Case Analysis**. Since the axioms have boolean preconditions, all the clauses generated in an I'-derivation are boolean. If an I'-derivation fails, then there exists a positive clause $C$ such that **Generation** cannot be applied to $C$. Moreover, for each equation $e_i$ in $C$, for all test substitutions $\sigma$ and for all context $c$ applicable to $e_i$, $c[e_i]\sigma$ does not match any left-hand side of $R$, otherwise conditional rewriting by $\rightarrow_R$ or **Case Analysis** can be applied to $c[e_i]\sigma$ since $R$ is strongly complete. So, for each test clausal context $c$ for $C$, $c[C]\sigma$ is not a tautology, and it is provably inconsistent. Then, **Disproof** is applied on $C$. By Theorems 6.2 and 6.4, we conclude that $R \not\models_{obs} E_0$. □

The following example is proposed in [Lys92].

**Example 6.1** *Consider variables of type nat built from* 0 *and s. The type var is built from init which denotes a variable initialized to* 0*, and from the ass which denotes the variable assignment operation. The function* value *takes a variable as argument and returns its value which is of sort nat. Suppose $S_{obs}$= {nat}. Let $R$ be the following rewriting system:*

$$\begin{aligned} value(init) &= 0 \\ value(ass(x, y)) &= y \end{aligned}$$

*A test context set is $\{z_{nat}, value(z_{var})\}$. A test set is $\{init, ass(x, y), x_{nat}\}$. Consider the following conjecture:*

$$ass(x, value(x)) = x$$

*Applying an induction, we obtain:*

$$\begin{aligned} value(init) &= value(init) \\ value(ass(x, y)) &= value(y) \end{aligned}$$

*These subgoals are simplified into tautologies. The initial conjecture is an observational theorem w.r.t. $R$. Note that it is possible to apply only test contexts to prove the initial conjecture, which allows the simplification of proofs in general. However, to refute, it is*

*necessary to apply at the same time test contexts and test substitutions, as it is the case for the following conjecture:*

$$ass(x, y) = init$$

*Applying an induction, we obtain:*

$$value(ass(x, y)) = value(init)$$

*It is simplified into:*

$$y = 0$$

*which is a provably inconsistent clause. Since $R$ is ground convergent, by Theorem 6.5 we conclude that the initial conjecture is not an observational theorem w.r.t. $R$.*

# 7 Computer experiments

We have implemented[2] these results in the Spike prover [BR95], written in Caml Light, and tested several examples.

## 7.1 Stack

We proved automatically that $push(top(S), pop(S)) = S$ is a behavioural property of the specification stack (see figure 1). Note that this example fails with the approach of [BH92], since it is not possible to compute automatically a set of *crucial contexts*: if two stacks have the same top they are not necessarily equal. In the approach of [Hen91], we have to introduce an auxiliary function $iterated\_pop : nat \times stack \rightarrow stack$ such that $iterated\_pop(n, s)$ iterates $n$ times $pop$. This is easy because $pop$ is unary. The function $iterated\_pop$ is defined by:

$$\begin{aligned} iterated\_pop(0, s) &= s \\ iterated\_pop(n + 1, s) &= iterated\_pop(n, pop(s)) \end{aligned}$$

Then, we have to prove the property for all contexts of the form $top(iterated\_pop(x, c[z_{stack}]))$. However, this schematization of contexts could be more complicated in case of a function of arity greater than two. So, this process seems to be not easy to automatize in general. In the approach of [MG94], this problem remains too.

Now, let us describe our proof. The prover computes first a test set for $R$ and the induction positions of functions, which are necessary for inductive proofs. It also computes a test context. These computation are done only once and before the beginning of the proof.

```
test set of R:
```

[2] implementation not completely finished

```
 -> elem = {0, s(x1)}
 -> stack = {Nil ; push(x1,x2)}

test contexts of R:

 -> stack = {pop(x1)}
 -> elem = {x1, top(x1)}

induction positions of functions:

 -> top : [[1]]
 -> pop : [[1]]

E0 = {push(top(x1),pop(x1)) = x1}
```

The prover applies an induction on the conjecture $E_0$: it applies all the possible contexts and instanciates the induction variables by all test substitutions.

```
Application of generate on:
    push(top(x1),pop(x1)) = x1 :
 1) Nil = pop(Nil) ;
 2) x2 = pop(push(x1,x2)) ;
 3) 0 = top(Nil) ;
 4) x1 = top(push(x1,x2))
```

The lemma obtained are then simplified.

```
E1 = {Nil = pop(Nil) ;
      x2 = pop(push(x1,x2)) ;
      0 = top(Nil) ;
      x1 = top(push(x1,x2))}

H1 = {push(top(x1),pop(x1)) = x1}

Delete  Nil = pop(Nil)
 it is subsumed by:pop(Nil) = Nil of R

Delete  x2 = pop(push(x1,x2))
 it is subsumed by:pop(push(x1,x2)) = x2 of R

Delete  0 = top(Nil)
 it is subsumed by:top(Nil) = 0 of R

Delete  x1 = top(push(x1,x2))
```

```
 it is subsumed by:top(push(x1,x2)) = x1 of R

E2 = {}

H2 = {push(top(x1),pop(x1)) = x1}

The initial conjectures are observationally valid in R
```

## 7.2 Lists

Consider the specification *list* in figure 3. The theorem $insert(x1, insert(x1, x2)) = insert(x1, x2)$ is automatically proved.

```
test set of R:

 -> nat = {0 ; s(x1)}
 -> list = {Nil ; insert(x1,x2)}
 -> bool = {False ; True}

test contexts of R:

 -> bool = {x1, in(x1,x2)}
 -> list = {x1, union(x1,x2)}

induction positions of functions:

 -> union : [[1]]
 -> in : [[2]]
 -> eq : [[1];[2]]

E0 = {insert(x1,insert(x1,x2)) = insert(x1,x2)}
```

The prover applies an induction on the conjecture $E_0$. It applies all the possible test contexts on $E_0$ : it applies *in* and performs a case analysis, and it applies *union* and simplifies the lemma obtained by $R$.

```
Application of generate on:
    insert(x1,insert(x1,x2)) = insert(x1,x2) :
 1) eq(x3,x1) = True => True = in(x3,insert(x1,x2)) ;
 2) eq(x3,x1) = False => in(x3,insert(x1,x2)) = in(x3,insert(x1,x2)) ;
 3) eq(x3,x1) = False, eq(x3,x1) = True ;
 4) insert(x1,insert(x1,union(x2,x4))) = union(insert(x1,x2),x4)
```

```
Delete  eq(x3,x1) = False => in(x3,insert(x1,x2)) = in(x3,insert(x1,x2))

E1 = {eq(x3,x1) = True => True = in(x3,insert(x1,x2)) ;
      eq(x3,x1) = False, eq(x3,x1) = True ;
      insert(x1,insert(x1,union(x2,x4))) = union(insert(x1,x2),x4)}

H1 = {insert(x1,insert(x1,x2)) = insert(x1,x2)}


Delete  eq(x3,x1) = True => True = in(x3,insert(x1,x2))
 it is subsumed by:eq(x1,x2) = True => in(x1,insert(x2,x3)) = True of R

E2 = {eq(x3,x1) = False, eq(x3,x1) = True ;
      insert(x1,insert(x1,union(x2,x4))) = union(insert(x1,x2),x4)}

H2 = {insert(x1,insert(x1,x2)) = insert(x1,x2)}
```

The prover can use other hypotheses and conjectures during the simplification. Here, it uses the hypothesis $H_2$ to rewrite a conjecture.

```
Simplification of:
    insert(x1,insert(x1,union(x2,x4))) = union(insert(x1,x2),x4) by H2:
    insert(x1,union(x2,x4)) = union(insert(x1,x2),x4)

E3 = {eq(x3,x1) = False, eq(x3,x1) = True ;
      insert(x1,union(x2,x4)) = union(insert(x1,x2),x4)}

H3 = {insert(x1,insert(x1,x2)) = insert(x1,x2)}

Simplification of:
    insert(x1,union(x2,x4)) = union(insert(x1,x2),x4) by R[H3 U E3]:
    insert(x1,union(x2,x4)) = insert(x1,union(x2,x4))


E4 = {eq(x3,x1) = False, eq(x3,x1) = True ;
      insert(x1,union(x2,x4)) = insert(x1,union(x2,x4))}

H4 = {insert(x1,insert(x1,x2)) = insert(x1,x2)}


Delete  insert(x1,union(x2,x4)) = insert(x1,union(x2,x4))

Application of generate on:
```

```
    eq(x3,x1) = False, eq(x3,x1) = True :
 1) eq(0,0) = True, True = False ;
 2) eq(s(x1),0) = True, False = False ;
 3) eq(0,s(x1)) = True, False = False ;
 4) eq(s(x2),s(x1)) = True, eq(x2,x1) = False

Delete  eq(s(x1),0) = True, False = False

Delete  eq(0,s(x1)) = True, False = False

E5 = {eq(0,0) = True, True = False ;
      eq(s(x2),s(x1)) = True, eq(x2,x1) = False}

H5 = {eq(x3,x1) = False, eq(x3,x1) = True ;
      insert(x1,insert(x1,x2)) = insert(x1,x2)}


Delete  eq(0,0) = True, True = False
 it is subsumed by:eq(x1,x1) = True of R

E6 = {eq(s(x2),s(x1)) = True, eq(x2,x1) = False}

H6 = {eq(x3,x1) = False, eq(x3,x1) = True ;
      insert(x1,insert(x1,x2)) = insert(x1,x2)}

Simplification of:
    eq(s(x2),s(x1)) = True, eq(x2,x1) = False by R[H6 U E6]:
    eq(x2,x1) = True, eq(x2,x1) = False

E7 = {eq(x2,x1) = True, eq(x2,x1) = False}

H7 = {eq(x3,x1) = False, eq(x3,x1) = True ;
      insert(x1,insert(x1,x2)) = insert(x1,x2)}

Delete  eq(x2,x1) = True, eq(x2,x1) = False
 it is subsumed by:eq(x3,x1) = False, eq(x3,x1) = True of H7

E8 = {}

H8 = {eq(x3,x1) = False, eq(x3,x1) = True ;
      insert(x1,insert(x1,x2)) = insert(x1,x2)}


The initial conjectures are observationally valid in R
```

# 8 Conclusion

We have presented an automatic procedure for proving observational properties in conditional specifications. The method relies on the construction of a set of *test contexts* which enables to prove or disprove conjectures. Under reasonable hypotheses, we have shown that the procedure is refutational compelete: each non observationally valid conjecture will be detected after a finite time. Future possible improvements would be the extension of the observation technique to terms and formulas.

# References

[BB91] G. Bernot and M. Bidoit. Proving the correctness of algebraically specified software: Modularity and observability issues. In *AMAST'91*, Lecture Notes in Computer Science, pages 216–239. Springer-Verlag, 1991.

[BBK91] G. Bernot, M. Bidoit, and T. Knapik. Observational approaches in algebraic specifications: a comparative study. Technical Report LIENS-91-6, Laboratoire d'Informatique de l'Ecole Normale Superieure, 1991.

[BBK92] G. Bernot, M. Bidoit, and T. Knapik. Towards an adequate notion of observation. In Bernd Krieg-Brückner, editor, *European Symposium on programming*, volume 589 of *Lecture Notes in Computer Science*, pages 39–55, Rennes, February 1992.

[BH92] M. Bidoit and R. Hennicker. How to prove observational theorems with lp. In U. Martin and J. Wing, editors, *Proc. of First International Workshop on Larch*. Springer-Verlag, 1992.

[BH93] B. Bauer and R. Hennicker. Proving the correctness of algebraic implementations by the ISAR system. In *DISCO'93*, volume 722 of *Lecture Notes in Computer Science*, pages 2–16. Springer-Verlag, 1993.

[BH96] M. Bidoit and R. Hennicker. Behavioural theories and the proof of behavioural properties. *TCS*, 165(1):3 – 55, 1996.

[BK89] K. Bundgen and W. Küchlin. Computing ground reducibility and inductively complete positions. In N. Dershowitz, editor, *Proceedings 3rd Conference on Rewriting Techniques and Applications, Chapel Hill (N.C., USA)*, Lecture Notes in Computer Science, pages 59–75. Springer-Verlag, 1989.

[Bou97] A. Bouhoula. Automated theorem proving by test set induction. *JSC*, 1997. To appear. Also available via http://www.loria.fr/ bouhoula/Bouhoula-JSC97.ps.

[BR95] Adel Bouhoula and Michaël Rusinowitch. Implicit induction in conditional theories. *Journal of Automated Reasoning*, 14(2):189–235, 1995.

[GGM76] V. Girranata, F. Gimona, and U. Montanari. Observability concepts in abstract data type specification. In *Proceedings 1st International Symposium on Mathematical Foundations of Computer Science*, volume 45 of *Lecture Notes in Computer Science*, pages 576–587. Springer-Verlag, 1976.

[Gut75] J. Guttag. *The specification and Application to Programming of Abstract Data Types*. Thèse de Doctorat d'Université, University of Toronto, 1975.

[Hen89] R. Hennicker. Implementation of parameterized observational specifications. In *TAPSOFT'89*, volume 351 of *lncs*, pages 290–305. Springer-Verlag, 1989.

[Hen91] R. Hennicker. Context induction: a proof principle for behavioural abstractions and algebraic implementations. *Formal Aspects of Computing*, 3(4):326–345, 1991.

[Kna91] T. Knapik. Specifications with observable formulae and observational satisfaction relation. *Recent Trends in Data Type Specifications*, 1991.

[KNZ87] D. Kapur, P. Narendran, and H. Zhang. On sufficient completeness and related properties of term rewriting systems. *Acta Informatica*, 24:395–415, 1987.

[Kou92] E. Kounalis. Testing for the ground (co-)reducibility property in term-rewriting systems. *Theoretical Computer Science*, 106:87–117, 1992.

[Lys92] O. Lysne. Proof by consistency in constructive systems with final algebra semantics. In *Proceedings 3rd International Conference on Algebraic and Logic Programming, Volterra (Italy)*, volume 632 of *Lecture Notes in Computer Science*, pages 276–290. Springer-Verlag, 1992.

[Lys94] O. Lysne. Extending Bachmair's method for proof by consistency to the final algebra. *Information Processing Letters*, 51:303–310, 1994.

[MG94] G. Malcolm and J. Goguen. Proving correctness of refinement and implementation. Technical Monograph PRG-114, Oxford University Computing Laboratory, November 1994.

[NO87] P. Nivela and F. Orejas. Initial behaviour semantics for algebraic specification. *Recent Trends in Data Type Specification*, 332:184–207, 1987.

[Pad88] P. Padawitz. *Computing in Horn Clause Theories*. Springer-Verlag, 1988.

[Pad96] P. Padawitz. Swinging data types: Syntax, semantics and theory. In *Recent Trends in Data Specifications, $11^{th}$ Workshop on Specification of Abstract Data Types*, volume 1130 of *Lecture Notes in Computer Science*, pages 409–435. Springer-Verlag, 1996.

[Rei84] H. Reichel. Behavioural validity of conditional equations. In *Contributions to General Algebra 3, Proc. of the Vienna Conference*, 1984.

[Sch90] O. Schoett. Behavioural correctness of data representation. *Science of Computer Programming*, 14:43–57, 1990.

[ST85] D. Sanella and A. Tarlecki. On observational equivalence and algebraic specification. In *TAPSOFT'85*, volume 185 of *Lecture Notes in Computer Science*, pages 308–322. Springer-Verlag, 1985.

[ST88] D. Sanella and A. Tarlecki. Toward formal development of programs from algebraic specification revisited. *Acta Informatica*, 25:233–281, 1988.

[ST89] D.T. Sanella and A. Tarlecki. Towards formal development of ml programs: foundations and methodology. In J. Diaz and F. Orejas, editors, *TAPSOFT'89*, volume 352 of *Lecture Notes in Computer Science*, pages 375–389. Springer-Verlag, 1989.

[Wan79] M. Wand. Final algebra semantics and data type extensions. *Journal of Computer and System Sciences*, 19:27–44, 1979.

[Wir90] Martin Wirsing. Algebraic specification. In J. van Leeuwen, A. Meyer, M. Nivat, M. Paterson, and D. Perrin, editors, *Handbook of Theoretical Computer Science*, volume B, chapter 13. Elsevier Science Publishers B. V. (North-Holland) and The MIT press, 1990.

Unit´e de recherche INRIA Lorraine, Technopˆole de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhˆone-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399